Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DCC205 - Estruturas de Dados
2º Semestre de 2023
Profs Eder e Wagner

Lista de exercícios

Questão 1 Indique se as afirmações a seguir são Verdadeiras ou Falsas e justifique.

1. Inserção é, em geral, mais eficiente que o Quicksort para arquivos quase ordenados.
2. O algoritmo de pesquisa em uma árvore AVL é o mesmo de uma árvore binária de pesquisa.
3. A complexidade de inserção em uma árvore AVL tem o pior caso igual a *O(n).*
4. O *Bucket Sort* tem complexidade $O(n^2)$ quando o número de *buckets* tende a n.
5. O *Counting Sort* requer uma memória auxiliar de tamanho n.
6. No pior caso, a pesquisa binária e a pesquisa sequencial têm a mesma ordem de complexidade.
7. O uso de um *sentinela* na pesquisa sequencial reduz a sua ordem de complexidade.
8. O melhor caso de uma pesquisa em tabela hash será O(1), independente do tratamento de colisão adotado.

Questão 2 Considere uma árvore AVL, onde remoções de chaves internas levam em consideração apenas o antecessor. Seja X seu número de matrícula % 100.

1. Insira as chaves X, 65, 68, 41, 19, 87, 92, 22, 76 e 3 na árvore. Qual a árvore resultante?
2. Insira as chaves 38, 24, 42, 78 e 61 na árvore. Qual a árvore resultante?
3. Remova as chaves 92, 22, 38 e 42 da árvore. Qual a árvore resultante?

Questão 3 Considere uma árvore B com até 4 chaves por nó, onde remoções de chaves internas levam em consideração apenas o antecessor. Empréstimos de registros na remoção apenas dos "irmãos", ou seja, nós que tenham o mesmo "pai". Seja X seu número de matrícula % 100.

1. Insira as chaves X, 22, 92, 24, 3, 67, 69, 78, 88, 38 e 87 na árvore e mostre o resultado.
2. Insira as chaves 89, 61, 76, 19, 41, 42 e 68 na árvore e mostre o resultado.
3. Remova as chaves 92, 67, 89, 41 e 22 da árvore e mostre o resultado.

Questão 4 Considere um hash de endereçamento aberto cuja função seja dada por h(i) = i mod 13. As colisões são tratadas de forma sequencial com passo igual a 1 (ou seja, considera a partir da próxima entrada da tabela a partir da posição indicada pelo hash). Diferencie as entradas não utilizadas daquelas onde houve remoção. Seja X seu número de matrícula % 100.

1. Insira as chaves X, 88, 42, 19, 3 e 87. Qual a configuração da tabela hash?
2. Insira as chaves 68, 22, 24, 65 e 38. Qual a configuração da tabela hash?
3. Remova as chaves 87, 22 e 24. Qual a configuração da tabela hash?

Questão 5 Você foi encarregado de realizar uma ordenação de um arquivo cujo conteúdo não cabe em memória primária usando ordenação polifásica com 4 "fitas" no total. Seja X seu número de matrícula % 100. Após uma primeira passada usando seleção por substituição, você obteve X+33 blocos para serem intercalados. Indique quantos blocos cada uma das fitas deve receber para que o arquivo ordenado seja posicionado na fita 1. Justifique a sua resposta.

Questão 6 O algoritmo do Quicksort, apesar de otimizado, ainda realiza uma quantidade significativa de movimentações de registros. O desempenho do algoritmo pode ser impactado por essas movimentações em função do tamanho dos registros. O código a seguir mostra as várias funções do algoritmo Quicksort e a definição da estrutura Item, que tem um tamanho significativo (8 megabytes em uma máquina 64 bits).

```
typedef struct {
  int Chave;
  int Conteudo [1000000];
} Item;
void Particao(int Esq, int Dir,
              int *i, int *j, Item *A){
  Item x, w;
  *i = Esq; *j = Dir;
  x = A[(*i + *j)/2];
  do {
     while (x.Chave > A[*i].Chave) (*i)++;
     while (x.Chave < A[*j].Chave) (*j)--;
     if (*i <= *j){
       w = A[*i];
       A[*i] = A[*j];
       A[*j] = w;
       (*i)++;
       (*j)--;
     }
   } while (*i <= *j);
}

void Ordena(int Esq, int Dir, Item *A){
  int i, int j;
  Particao(Esq, Dir, &i, &j, A);
  if (Esq < j) Ordena(Esq, j, A);
  if (i < Dir) Ordena(i, Dir, A);
}

void QuickSort(Item *A, int n){
  Ordena(0, n-1, A);
}
```

Uma estratégia para minimizar esse impacto é o chamado Quicksort indireto, onde estrutura Item é desmembrada em duas estruturas, uma contendo a Chave e outra contendo o Conteudo (conforme o exemplo). Reescreva o código apresentado (incluindo definição de estruturas) implementando uma estratégia indireta e minimizando o impacto das movimentações. Explicite quaisquer premissas que você tenha levado em consideração na sua proposta.

Questão 7 Para cada afirmação a seguir, indique se ela é verdadeira ou falsa e justifique sua resposta:

a)( ) Qualquer algoritmo de ordenação realiza pelo menos $\Omega(n.logn)$ operações.

b)( ) Não conhecemos nenhum algoritmo de ordenação cujo pior e o melhor caso sejam idênticos em termos de complexidade assintótica.

c)( ) Insertionsort e Selectionsort são exemplos de algoritmos de ordenação estáveis.

d)( ) QuickSort com escolha do pivô realizada através de sorteio, com todas as posições do vetor de entrada podendo ser escolhidas com mesma probabilidade, é um algoritmo ótimo de ordenação

e)( ) Um algoritmo de ordenação que realiza comparações e tem complexidade de $O(n.logn)$ é um algoritmo ótimo.

Questão 8 Seja a busca ternária uma extensão do algoritmo de busca binária onde o vetor é dividido em 3 partes ao invés de 2. Tendo isso em mente:

a)Complete o código a seguir que implementa a busca ternária:

```
int ternaria(int x, int * T, int Esq, int Dir){
   int j = (Esq+Dir)/3; if (j<Esq) j=Esq;
   int k = 2*(Esq+Dir)/3; if (k>Dir) k=Dir;
   if (Esq==Dir) return Esq;
   if (Dir-Esq>=2){
      if (T[___] >= x){
            if (T[___] == x) return ___;
            else return ternaria(x,T,_____);
      } else {
        if (T[___] >= x){
            if (T[___] == x) return ___;
            else return ternaria(x,T,_____);
        } else return ternaria(x,T,_____);
      }
   } else {
      if (T[___] == x) return ___;
      if (T[___] == x) return ___;
      return -1;
   }
}
```

b) Qual a complexidade do algoritmo em a)? Ele é assintoticamente melhor que a busca binária comum?

Questão 9 Vamos supor que temos um arranjo de *n* registros de dados para ordenar e que a chave de cada registro tem valor 0 ou 1. Um algoritmo para ordenar tal conjunto de registros poderia ter algum subconjunto das três critérios desejáveis a seguir:

1. O algoritmo é executado com complexidade de O(n).
2. O algoritmo é estável.
3. O algoritmo ordena localmente, sem utilizar mais que uma quantidade constante de espaço de armazenamento além do arranjo original.

a)Indique um algoritmo que satisfaça os critérios 1 e 2.
b)Indique um algoritmo que satisfaça os critérios 1 e 3.
c)Indique um algoritmo que satisfaça os critérios 2 e 3.
d) Algum dos seus algoritmos de ordenação dos itens (a)-(c) pode ser usado para ordenar *n* registros com chaves de *b* bits usando *radix sort* com complexidade O(*bn*)? Explique como ou por que não.

Questão 10 Considere uma árvore B com até 4 chaves por nó, onde remoções de chaves internas levam em consideração apenas o antecessor. Empréstimos de registros na remoção apenas dos "irmãos", ou seja, nós que tenham o mesmo "pai" e sejam adjacentes aos nós que solicitam o empréstimo. Seja X seu número de matrícula % 100.

1. Insira as chaves X, 54, 105, 56, 27, 5, 35, 94, 101, 86, 88, 102 e 78 na árvore e mostre a árvore resultante.
2. Insira as chaves 100, 69, 96, 103, 59, 58 e 75 na árvore do item 1 e mostre a árvore resultante.
3. Remova as chaves 35, 88, 102, 69 e 103 da árvore do item 2 e mostre a árvore resultante.